Amendments to the Specification:

Please replace the first full paragraph on page 3 with the following amended paragraph:

A1

In a main-memory DBMS, for example, where the entire database is kept in main memory, disk access for logging acts as a bottleneck in the system performance. In order to reduce such a bottleneck, employment of multiple ~~log disks~~ persistent log storage devices may be conceived to distribute the processing. The use of multiple ~~log disks~~persistent log storage devices, however, is not easily amenable to the conventional logging method because there is necessarily an overhead of merging log records in the order of creation during the step of LP.

Please replace paragraphs 9-11 on page 4 with the following amended paragraphs:

N.E.

FIG. 12 a diagram of one embodiment of a logging architecture of the present invention where multiple ~~log disks~~ persistent log storage devices are used to distribute log records and to enable parallel logging.

FIG. 13 is a diagram of one embodiment of a restart architecture of the present invention where multiple ~~log disks and backup disks~~ persistent log storage devices and persistent backup storage devices are used to enable parallel restarting.

A2

FIG. 14 is a flow chart of a fully parallel restart process used in the present invention to recover the database as quickly as possible using multiple ~~log disks and backup disks~~persistent log storage devices and persistent backup storage devices.

Please replace the second paragraph on page 5 with the following amended paragraph:

A3

An update log record is used to store changes in the database comprises of a log header and a log body. FIG. 4 shows a variety of fields in the log header according to a preferred embodiment. The "LSN (Log Sequence Number)" field stores the identity of the current log record by preferably storing the physical address of log record on disk. The "TID (Transaction ID)" field stores the identity of the transaction associated with the current log record. The "PrevLSN" field stores the identity of the log record that was most recently created by the same transaction so that the information can be conveniently used to chain the log records of a transaction for fast backward retrieval. The "Type" filed stores the type of log record. The "Backup ID" field stores the relation between the log record and the changed page. This field needs to be maintained only when using a fuzzy checkpointing scheme, which will be explained later in FIGs. 8a and 8b. The "Page ID" field stores the identity of a page where the update occurred. The "Offset" field stores the identity of a slot inside a page where the update occurred. The "Size" field stores the length of the updated slot.

Please replace lines 5-34 on page 9 and 1-4 on page 10 with the following amended paragraphs:

FIG. 12 shows an embodiment of a logging architecture of the present invention where multiple ~~log disks~~persistent log storage devices are used to distribute log records and to enable parallel logging. Unlike the physical logging scheme, the differential logging scheme allows us to freely distribute log records to multiple disks to improve the logging performance because the commutativity and associativity of XOR used in the differential logging scheme enables processing of log records in an arbitrary order. In this embodiment, log records are distributed to multiple ~~log disks~~persistent log storage devices based on transaction identifiers (TIDs). There is a different log manager (1201) for each ~~log disk~~persistent log storage device (1203). When a transaction starts, the identity of the log manager (LogMgr ID) having the least amount of load is assigned to the transaction so that the log records generated by the transaction are always stored in the ~~log disk~~persistent log storage device.

FIG. 13 shows an embodiment of a restart architecture of the present invention where multiple ~~log disks and backup disks~~ persistent log storage device and persistent backup storage devices are used to enable parallel restarting. As mentioned in FIG. 2, the restart process comprises the four sub-processes, BR, BP, LR, and LP. In this embodiment, three types of parallelism are utilized to speed up the restart process. First, a different backup loader (BL)

(1302) is instantiated for each ~~backup disk~~persistent backup storage device (1307), and it performs the BR process (1302) and the BP process (1301) in a pipeline manner. Each BL runs independently of other BLs. Second, a different log loader (LL) (1305) is instantiated for each ~~log disk~~persistent log storage device (1308), and it performs the LR process (1306) and the LP process (1304) in a pipeline manner. Each LL runs independently of other LLs. Third, even BLs and LLs are run concurrently.

FIG. 14 is a flow chart of a fully parallel restart process used in the present invention to recover the database as quickly as possible using multiple ~~log disks and backup disks~~persistent log storage device and persistent backup storage device. First, all the pages in the database are filled with 0s (1401). Since 0 is the identity of the XOR operation, this allows the BP process to use the XOR operation to restore a page in the database instead of copying a page into the database. Second, one BL module and one LL module are created for each ~~backup disk and log disk~~persistent backup storage device and persistent log storage device (1402 and 1403), respectively. They run independently and in parallel.

FIG. 15 is a diagram of one embodiment of the backup loader (BL) module used in the fully parallel restart process of the present invention. The BL runs the BR process (1504) and the BP process (1502) in a pipeline manner or in the producer/consumer way. The BR process reads the ~~backup disk~~persistent backup storage device (1505) page by page and appends each page to the page buffer (1503), which is used like a pipeline. The BP process takes a page from

A13

the buffer and applies it to the matching page in the database (1501). A novel aspect of the present invention is that the BP process uses the XOR operation when applying a backed-up page to the database. This allows us to run BLs and LLs concurrently, which was not possible in the conventional art.

Please replace the third full paragraph on page 10 with the following amended paragraph:

A14
A15

In this embodiment, each log buffer page has a counter. This counter is reset when the LR process (1704) fills it with the data read from the ~~log disk~~ persistent log storage device (1705). When an LP process finishes scanning a buffer page, it increment the counter of the page in a locked state. Then, when the counter has the same value as the number of LP processes, the buffer can be flushed.
